VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the company

2014 Jiří Arleth

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jiří Arleth**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: ABB s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Kot, Ph.D.**

Konzultant bakalářské práce: Bc. Jiří Gargaš

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 2014 ….....

Podpis

Poděkování

Mé poděkování patří Bc. Jiřímu Gargašovi za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování bakalářské práce věnoval. Dále bych rád poděkoval Ing. Martinu Kotovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování této práce. Nemenší díky pak také patří Andree Pravdové za pomoc při gramatické kontrole práce.

Abstrakt

Tato bakalářská práce je shrnutím průběhu mé bakalářské praxe u společnosti ABB s.r.o. Stručně představí firmu a mou pozici. Hlavním obsahem je popis úkolů, jež jsem plnil. Ty se týkaly především testování software, hlavně funkčního testování, který probíhal formou Coded UI testů od Microsoftu.

Klíčová slova

ABB, CUIT, Testování, C#, .NET, Microsoft Visual Studio

Abstract

This bachelory thesis is a summary of my experiences during bachelory practice by the ABB inc. company. The company is briefly introduced as well as my employment. Main topic is about given tasks, mostly software testing, especially functional testing. Most attention is dedicated to the Microsoft Coded UI Tests. The conclusion contemplates about the whole practice and brings some interesting facts and thoughts.

Keywords

ABB, CUIT, Testování, C#, .NET, Microsoft Visual Studio

Seznam použitých zkratek a symbolů

CZOPC – Operační centrum Česká republika

CUIT – Coded User Interface Test

VS – Visual Studio

VS2012 – Visual Studio 2012

SW - software

API – Application Programming Interface

GUI – Graphical User Interface

CSV – Comma-separated Values

OPC - OLE for Process Control

DGMS – Diesel Generator Monitoring System

PMS – Power Management System

WPF – Windows Presentation Foundation

Obsah

Seznam použitých obrázků

Seznam ukázek kódu

1. Úvod

Tato práce se věnuje popisu mé bakalářské praxe ve firmě ABB. Nejprve je zde krátké seznámení se společností, osvětlení mé pozice a úkolu. Tím byla příprava testovacího prostředí pro simulátor lodních systémů pro norského zákazníka ABB.

Prvně tedy bylo nutné zjistit si a nastudovat potřebnou teorii k samotnému testování softwaru, následně pak přistoupit k hledání vhodného řešení a nakonec k jeho implementaci.

Ta probíhala v jazyce C# a mým nalezeným řešením byly Coded UI Testy, tedy automatizované funkční testy.

V zakončení pak popisuji výsledky a přínos praxe pro mou osobu a užitečnost a zúročení znalostí získaných na škole a co naopak chybělo.

2. O společnosti ABB

ABB (Asea Brown Boveri) je švédsko-švýcarská nadnárodní korporace se sídlem v Curychu, působící zejména na poli technologií automatizace a energetiky s účelem zvýšení produkce zákazníků a zároveň snížení dopadu jejich činností na životní prostředí. Zaměstnává více než 150 000 zaměstnanců ve 100 zemích a v roce 2010 měla obrat 31,6 miliard dolarů. V Česku společnost zaměstnává téměř 3 300 zaměstnanců. Viz [1].

2.1 Oblasti podnikání

ABB je rozdělena do pěti divizí:

1. Výrobky pro energetiku (Power Products)

Tato divize se zabývá výrobou a dodávkami transformátorů, rozváděčů, vypínačů, a souvisejících zařízení.

2. Systémy pro energetiku (Power Systems)

Divize poskytuje dodávky systémů a služeb na klíč pro přenosové a distribuční sítě a elektrárny.

3. Automatizace výroby a pohony (Discrete Automation and Motion)

Divize poskytuje výrobky, řešení a s nimi související služby, které zvyšují průmyslovou produktivitu a energetickou účinnost.

4. Výrobky nízkého napětí (Low Voltage Products)

Divize Výrobky nízkého napětí vyrábí produkty, které zajišťují bezpečnost osob a ochranu instalací a elektrických přístrojů před přetížením.

5. Procesní automatizace (Process Automation)

Hlavním cílem této divize je poskytnout zákazníkům výrobky a řešení pro instrumentaci, automatizaci a optimalizaci průmyslových procesů.

Obrázek 1 – Sídla ABB v České republice

2.2 Mé zařazení

Svou praxi jsem vykonával v ostravském CZOPC umístěném v areálu průmyslové zóny Ostrava-Hrabová, které se koncem března přestěhovalo do Nové Karoliny v centru města. Působil jsem na pozici Software tester/developer. CZOPC je součástí divize Procesní automatizace.

3. Zadané úkoly a jejich řešení

Hlavním cílem celé mé praxe bylo nalézt řešení pro funkční testování obecně. To mělo následně být prakticky využito a ozkoušeno na konkrétní aplikaci, s níž jsem po celou dobu pracoval. Jednalo se o ABB Norway Marine Simulator, jehož hlavním účelem je simulace běhu lodních systémů. Aplikace je v beta verzi a je již sice v provozu a používána; kvůli chystanému refactoringu a velkým strukturálním úpravám však bylo třeba utvořit silnou a pevnou základnu z unit a funkčních testů, aby se tyto budoucí procesy mohly provádět v duchu Test-Driven Development, tedy testy řízeného vývoje. Cílem bylo těmito testy pokrýt minimálně 80% celkového kódu aplikace.

3.1 Studium testů

Jelikož během studia nebylo testování vůbec probíráno, musel jsem si ještě před výběrem vhodné technologie k funkčnímu testování ujasnit, co se funkčními testy myslí. Jak probíhají, jaký je jejich smysl a účel [3, 4, 7].

3.1.1 Kategorie testů

Testy lze dělit podle několika kritérií:

1. Statické a dynamické

Toto je dělení na základě toho, zda je nutné k spuštění testu spustit i aplikaci. Statické testy toto nevyžadují, dynamické ano.

2. Black box a white box

V překladu „testování černé či bílé skříňky“ znamená, zda-li je pro tvorbu testů nezbytné disponovat znalostmi o vnitřní skladbě a fungování testovaného softwaru či nikoliv. Jinými slovy: buď do softwaru vidíme nebo ne. Proto také ty výstižné názvy. U black box pouze definujeme vstupy a kontrolujeme výstupy. Co se děje uvnitř, nevidíme. Při testování white box má tester přístup k zdrojovému kódu, a tak vidí i vnitřní reakce systému. Krom těchto dvou metod, existuje také gray box testing, což je jejich kombinace.

3. Automatické a manuální

Buď musí být testy prováděny ručně testerem, anebo automaticky. Zatímco první způsob je vhodnější zejména během rané fáze vývoje softwaru, kdy se neustále řeší nové a nové případy, automatizace se využívá při opakovaném spouštění velkého množství testů nebo u testů s velkým počtem generovaných dat.

3.1.2 Hierarchie testů

1. Unit testy

Testy prováděné programátorem. Sám si kontroluje metody svých objektů a funkčnost skriptů.

2. Funkční testy

Ověřování funkčnosti SW vzhledem k požadavkům zákazníka.

3. Integrační testy

V této fázi se testuje integrace dosud jednotlivě ověřených částí SW.

4. Akceptační testy

Ověřují, že aplikace splňuje všechny zákazníkovy požadavky. Často jsou prováděny přímo klientem.

5. Regresní testy

Akceptace nemusí znamenat úplný konec vývoje aplikace. Může docházet k rozšiřování funkcí aplikace a v těchto případech je nutné se ujistit, že neexistují odchylky mezi výstupem před zásahem do aplikace a výstupem po zásahu.

Po nastudování principu funkčních testů, což zabralo necelé dva dny, mi vyšlo, že bude třeba nalézt testovací nástroj, který by umožňoval psát testy typu black box, dynamické a automatické. Prvním takovým pokusem byl nástroj Fitnesse [2].

3.2 Seznámení s Fitnesse

Jedná se o testovací framework, kde se testy vytvářejí formou wiki stránek. Do C# projektu bylo nejprve nutno přilinkovat příslušné knihovny, následně utvořit třídu dědící z fit.fixture. Na názvy třídy a metod se pak lze odkazovat ve wiki kódu, který je veskrze intuitivní a přirozený.

Jako první musí být řádek určující typ kódu, jelikož syntaxí je vícero. Já jsem používal ActionFixture. (Další jsou např. RowFixture, ColumnFixture, TableFixture, DoFixture...)

Každý další řádek ActionFixture je uveden příkazovou buňkou a následován buňkami s příslušnými argumenty. Příkazy jsou čtyři:

- **start** – očekává jeden argument, kterým je název třídy jež má být použita k testování
- **check** – přijímá dva parametry. Prvním je název metody*, která se spustí, druhým očekávaný výstup
- **press** – jediným parametrem je název void metody, která se provede
- **enter** – první argument je opět název metody*, jež se má spustit, následují hodnoty jejích vstupních parametrů
- pro .NET je možno těmito příkazy mimo metod přistupovat i k atributům a property

Zde je ukázka kódu třídy, která provádí test, kde jde o spojení dvojice řetězců oddělených čárkou:

```
using System;

namespace info.fitnesse.fixturegallery
{
    public class ActionFixtureTest : fit.Fixture
    {
        public String firstPart, secondPart, together;
        public void join()
        {
            together = firstPart + ", " + secondPart;
        }
    }
}
```

Ukázka kódu 1 – Názorná podoba třídy k testování ve Fitnesse frameworku

A následuje wiki kód:

```
!|ActionFixture|              // Používám ActionFixture syntaxi
|start|ActionFixtureTest|     // Budu používat třídu ActionFixtureTest
|enter|firstPart|Hello|       // Nastav atribut firstPart na hodnotu Hello
|enter|secondPart|World|      // Nastav atributu secondPart hodnotu World
|press|join|                  // Proveď metodu join(), která spojí oba řetězce
|check|together|Hello, World| // Má atribut together hodnotu „Hello, World“?
```

Ukázka kódu 2 – Příklad wiki kódu ve Fitnesse frameworku

Framework pak následně vygeneruje tabulku, kde je přehledně shrnut výsledek testu.

Přestože na první pohled se užití tohoto frameworku zdálo býti ideální, nešlo přesně o to, co jsem hledal. K testování aplikace bych vždy musel pro daný test utvořit samostatnou třídu, jejíž metody by přímo přistupovaly do logické vrstvy aplikace, což bylo nežádoucí, protože mělo jít o funkční testy. Tedy typ black box a testování skrze prezentační vrstvu.

Instalace, zprovoznění, seznámení se a ozkoušení Fitnesse si vyžádalo celkově tři dny práce.

3.3 Seznámení s Microsoft Coded UI Tests

Cituji popis fungování a zaměření CUITs, tak jak je to popsáno na [6]:

„Automated tests that drive your application through its user interface (UI) are known as coded UI tests (CUITs). These tests include functional testing of the UI controls. They let you verify that the whole application, including its user interface, is functioning correctly.“

Jak vidno, CUITs jsou přesně tím, co jsem hledal. Automatizují testování skrze uživatelské rozhraní vaší aplikace, tzn. není zde pro programátora žádná potřeba znalosti vnitřního kódu. Takže v tomto případě se již opravdu jedná o funkční testování, přesně jak bylo vyžadováno. Když k tomu navíc přidáme, že tento nástroj je od Microsoftu a přímo součástí Visual Studia, nebylo již co řešit. Viz [6].

3.4 Příprava prostředí

Na rozdíl od Fitnesse, kde bylo ke zprovoznění potřeba provést několik desítek kroků (pro zajímavost jsem je sepsal do textového souboru Fitnesse.txt v příloze), Coded UI Testy potřebují ke svému fungování pouze Microsoft Visual Studio 2012 Premium nebo Ultimate s nainstalovaným Update 1 (Visual Studio 2010 již neberu na zřetel, ale tam byly CUITs poprvé představeny).

Po nainstalování těchto prerekvizit již tedy bylo vše připraveno a ta skutečná programátorská práce mohla začít. Spolu s předchozím úkolem bylo toto provedeno za jeden den.

3.5 Tvorba testů

V této fázi již došlo na skutečnou programátorskou práci. Nejprve bylo nutné se pořádně seznámit s aplikací. Zejména s její strukturou a architekturou. Následně jsem dostal seznam scénářů, dle kterých bylo třeba naprogramovat Coded UI Testy. Když se posléze ukázalo, že tyto testy dokázaly pokrýt pouze 70% kódu, musel jsem přijít s vlastními scénáři. Tímto bylo dosaženo pokrytí 77%. Zbývající tři procenta pak vzešla z unit testů, které jsem dopsal pro některé z klíčových a často používaných tříd.

Tuto část práce mi však nebylo dovoleno zveřejnit, čili její podrobný popis a průběh je obsažen v neveřejné příloze. Zde uvedu pouze seznam názvů všech dílčích úkolů této fáze a počet dní, které jsem nad nimi strávil.

CUIT nahrávání konfiguračních CSV souborů a Simulinkových modelů	2
CUIT vykonávání simulace	2
CUIT komunikace s OPC serverem a Simulinkem	2
CUIT nahrávání simulace a export do Excelu	2
CUITs tvorby automatizovaných simulací v modulu Testing Center	3
CUIT možnosti ukládání a spouštění simulací ve frontě v modulu Testing Center	1
CUIT konfigurace aplikace	1
CUITs základních komponent systémů DGMS a PMS	4
CUITs složených komponent systémů DGMS a PMS	4
Oprava stávajících nefungujících unit testů	1
Unit testy třídy SimConvert	1
Unit testy třídy UserLog	1
Převod všech unit testů z NUnit na MS	2

3.6 Prezentace CUITs

Během mé praxe jsem byl svým konzultantem požádán abych představil Coded UI Testy ostatním kolegům na oddělení, jelikož se tato forma testování ukázala být velmi užitečná a efektivní. Připravil jsem tedy prezentaci a seznámil ostatní s možnostmi a funkčností této technologie. Prezentoval jsem dohromady dvakrát. Jednou v českém a podruhé v anglickém jazyce. CUITs se poté uchytily a začaly se v SW oddělení CZOPC používat. Byly začleněny do procesu vývoje všech nových softwarů.

Shromažďování potřebných informací a příprava prezentace vyšly na dva dny práce. Prezentace a mnou psaný manuál CUIT Introduction pro tvorbu Coded UI Testů jsou k dispozici v příloze na CD [5].

3.7 Refaktorování

Když jsem byl hotov s testováním, došlo na tento úkol. Na vývoji Simulatoru se již v minulosti vystřídalo asi pět programátorů, každý s odlišnými zkušenostmi a jiným „rukopisem“. Nyní vznikl požadavek, aby došlo k ujednocení kódu hlavně co se formální stránky týče – názvy proměnných, komentáře, členění do regionů, odsazování – ale také ze stránky architektonické. Jádro aplikace (jediná třída) čítalo přes 3000 řádků kódu! Bylo tedy třeba oddělit příslušné funkcionality do samostatných tříd podle návrhového vzoru Object Model. Za pět dní se mi podařilo zkrátit jádro na 1100 řádků, což už je příjemnější číslo.

3.8 Dokumentace

V průběhu praxe došlo ke změně Product Ownera (osoba na straně zákazníka, s níž se o Simulatoru jednalo) a bylo nutné jej do všeho zasvětit. Tudíž jsem vypracoval dokument popisující základní aspekty produktu (v podstatě uživatelskou příručku) a programátorskou dokumentaci v aplikaci DoxyGen, se kterou jsem již pracoval i na škole v rámci předmětu ZP (Základy programování). Celkový čas: pět dní.

3.9 Řešení bugů

Díky mým testům bylo odhaleno nemalé množství bugů a chybiček. Jednalo se především o věci spíše kosmetického rázu v rámci GUI, našlo se však i pár skutečně závažných problémů, např. chybně napsané unit testy pro komunikaci s OPC serverem [8], které vždy hlásily selhání, nehledě na okolnosti.

Další chyby nám hlásila zákaznická strana. Za všechny jmenuji tu nejzávažnější: havarování aplikace po více než hodinovém nepřetržitém běhu simulace. Jelikož v terénu se tento scénář objevuje jen vzácně, doposud tato závada zůstávala skryta. Její původ byl přitom velice prostý – špatně parsovaný textový řetězec při výpisu aktuálního času simulace do GUI.

Řešení bugů probíhalo vždy teprve když se nějaký objevil. Dohromady jsem tímto ale strávil tři dny.

3.10 Výpomoc na projektu RDS Commissioning Tool

V jednu chvíli jsem byl požádán konzultantem o výpomoc s jeho projektem RDS Commissioning Tool. Dostal jsem pouze požadavky na pár metod – vstupy a výstupy. Seznámení s projektem nebylo třeba. Pro představu uvedu dvě nejzajímavější metody.

1. Metoda pro tvorbu zástupce

Metoda měla na vstupu dostat cestu k originálnímu souboru (desktopová aplikace), vytvořit na ploše zástupce a vrátit true. V případě selhání false.

K tvorbě zástupců slouží knihovna `Windows Script Host Object Model`. Tu bylo nutné ve Visual Studiu přidat do referencí k projektu a následně šlo využít třídy `WshShell` a rozhraní `IWshShortcut`, což bylo vše potřebné.

```
/// <summary>
/// Creates shortcut on desktop to the given file.
/// </summary>
/// <param name="path">Path to file the shortcut should refer to.</param>
/// <returns>True on success, false otherwise.</returns>
public static bool AddShortcutToDesktop(string path)
{
    try
    {
        if (System.IO.File.Exists(path))
        {
            var desktop = Environment.GetFolderPath(Environment.SpecialFolder.DesktopDirectory);
            var fileName = Path.GetFileName(path);

            WshShell shell = new WshShell();
            IWshShortcut shortcut = (IWshShortcut)shell.CreateShortcut(Path.Combine(desktop,
                                        new FileInfo(fileName).Name + ".lnk"));

            shortcut.Description = fileName; // The description of the shortcut
            shortcut.TargetPath = path;      // The path of the file that will launch when the shortcut
                                                is run
            shortcut.Save();                 // Save the shortcut
            return true;
        }
    }
    catch (Exception e)
    {
        return false;
    }
    return false;
}
```

Ukázka kódu 3 – Metoda AddShortcutToDesktop

2. Metoda pro změnu typu spouštění služby

Tato metoda měla u dané služby změnit typ spouštění na požadovaný typ. Tedy Automaticky/Ručně/Zakázáno/Zpožděné spuštění

Zde bylo nezbytné využít metod z nativní knihovny Windows API `advapi32.dll`. Na generování hlaviček těchto metod pro C# jsem objevil užitečný doplněk k Visual Studiu – PInvoke [9] – který je zcela zdarma a snadno použitelný. Pro typy spouštění jsem si vytvořil výčtový typ parafrázující příslušné konstanty.

```
enum ServiceStartType {
        AUTO_START = 0x00000002,
        BOOT_START = 0x00000000,
        DEMAND_START = 0x00000003,
        DISABLED = 0x00000004,
        SYSTEM_START = 0x00000001
}
```

Ukázka kódu 4 – Výčtový typ ServiceStartType

Importováním a užitím metod z Windows API šlo pak snadno se službami pracovat. Jediným zádrhelem byla přístupová práva, což se vyřešilo použitím správné konstanty v metodách `OpenService` a `OpenSCManager`.

```
private const int SC_MANAGER_ALL_ACCESS = 0xF003F;
private const uint SERVICE_NO_CHANGE = 0xFFFFFFFF;

[DllImport("advapi32.dll", EntryPoint = "OpenSCManagerW", ExactSpelling = true, CharSet =
CharSet.Unicode, SetLastError = true)]
public static extern IntPtr OpenSCManager(string machineName, string databaseName, uint dwAccess);

[DllImport("advapi32.dll", SetLastError = true, CharSet = CharSet.Auto)]
static extern IntPtr OpenService(IntPtr hSCManager, string lpServiceName, uint dwDesiredAccess);

[DllImport("advapi32.dll", CharSet = CharSet.Unicode, SetLastError = true)]
public static extern Boolean ChangeServiceConfig(IntPtr hService, UInt32 nServiceType, UInt32
nStartType, UInt32 nErrorControl, String lpBinaryPathName, String lpLoadOrderGroup, IntPtr lpdwTagId,
String lpDependencies, String lpServiceStartName, String lpPassword, String lpDisplayName);

/// <summary>
/// Changes start type of the given service.
/// </summary>
/// <param name="serviceName">Service name.</param>
/// <param name="startType">Start type.</param>
/// <returns>True on success, false on failure.</returns>
public static bool ChangeServiceStatus(string serviceName, ServiceStartType startType)
{
    try
    {
        var ptr = OpenService(OpenSCManager(null, null, SC_MANAGER_ALL_ACCESS), serviceName,
                SC_MANAGER_ALL_ACCESS);
        ChangeServiceConfig(ptr, SERVICE_NO_CHANGE, (uint)startType, SERVICE_NO_CHANGE, null, null,
                        IntPtr.Zero, null, null, null, null);
        return true;
    }
    catch (Exception e)
    {
        return false;
    }
}
```

Ukázka kódu 5 – Metoda ChangeServiceStatus

Za další metody, které jsem inplementoval zmíním např. metodu na pořizování screenshotů nebo metodu pro přepisování registru. Celkově jsem na tomto projektu RDS Commisioning Tool pomáhal po dobu tří dnů.

4. Znalosti a dovednosti

V průběhu mé praxe došlo na mnoho činností, které více či méně souvisely s učivem na škole. Zde popíšu, na které z nich jsem byl v průběhu tříletého studia připraven a na které nikoliv.

4.1 Uplatněné znalosti a dovednosti

Zdaleka nejužitečnějším předmětem, ač se to může zdát překvapivé, se v průběhu mé praxe ukázala být angličtina. Veškerá komunikace se zákazníky, nadřízenými i kolegy ve firmě probíhala v naprosté většině případů v angličtině. Jazyk jsem zúročil rovněž při prezentování Coded UI Testů firmě.

Mezi další velice bohatě využité schopnosti pak samozřejmě patří programování. V tomto ohledu tedy patří můj dík veškerým předmětům zabývajícím se touto tematikou, především pak těm o objektovém programování a C#, čili PJ1 a PJ2.

Poslední věcí, jež jsem mohl zužitkovat, byly znalosti o trojvrstvé architektuře systémů z předmětu Vývoj informačních systémů, a pak předmět Softwarové inženýrství, především učivo týkající se agilní metody vývoje softwaru SCRUM.

4.2 Chybějící znalosti a dovednosti

Pokud mi něco chybělo, tak určitě jakékoli znalosti o testování softwaru. Toto učivo se za celou dobu mého studia neobjevilo ani na malou chvíli. V tomto ohledu jsem byl tedy jaksi vržen do vody a musel se naučit plavat.

Na druhou stranu jsem však díky tomu byl nucen věnovat poměrně mnoho volného času samostudiu této problematiky, což by mělo být podstatou a smyslem bakalářských prací – získání nových poznatků a vědecké bádání.

5. Závěr

Jen těžko bych si dokázal představit téma bakalářské práce, jež by dokázalo tak komplexním způsobem zúročit a prohloubit znalosti získané v průběhu oněch tří let studia. Praktické užívání nabytých znalostí a dovedností a následné získávání cenných zkušeností v oboru je tím nejhodnotnějším přínosem pro jakéhokoliv budoucího absolventa, kterýžto pak zajisté bude i disponovat většími šancemi na pracovním trhu.

Konkrétně jsem se tedy naučil vývoj WPF aplikací, tvorbu unit a Coded UI testů. Ve firmě se rovněž hodně dbalo na dodržování formálních konvencí, takže jsem si je díky tomu osvojil. Nesmím opomenout ani rozvoj mých soft-skills, jelikož bylo třeba aktivně komunikovat se zákazníkem o každém z mých kroků a účastnit se týmových SCRUM meetingů.

Za sebe hodnotím tuto praxi pozitivně v každém směru a vezmu-li v potaz spokojenost zákazníka, mohu rovněž považovat tuto praxi i za úspěšnou a přínosnou pro firmu.

6. Reference

[1] ABB - technologie pro energetiku a automatizaci. Dostupné na WWW: <http://www.abb.cz> [cit. 2014-02-16].

[2] Fitnesse - The fully integrated standalone wiki and acceptance testing framework. Dostupné na WWW: <http://fitnesse.org/> [cit. 2014-03-02].

[3] SW Testování. Dostupné na WWW: <http://www.swtestovani.cz/> [cit. 2014-02-23].

[4] Testování softwaru - Portál zabývající se problematikou ověřování kvality software, manuální i automatizované testování. Dostupné na WWW: <http://testovanisoftwaru.cz/> [cit. 2014-02-23].

[5] Dostupné v příloze: ARLETH, Jiří. ABB, s.r.o. CUIT Introduction. Ostrava, 2013.

[6] Microsoft Developer Network - Verifying Code by Using UI Automation. Dostupné na WWW: <http://msdn.microsoft.com/en-us/library/dd286726.aspx> [cit. 2014-03-22].

[7] Prezentace: VEPŘEK, Tomáš. Tieto Corporation Introduction to software testing. Ostrava, 2012

[8] Reliance – Industrial SCADA/HMI system. Dostupné na WWW: <http://www.reliance.cz/cs/products/opc-servers> [cit. 2014-04-25].

[9] Pinvoke.net – The Interop Wiki. Dostupné na WWW: <http://www.pinvoke.net/> [cit. 2014-07-05]